Kapitel 2: Intelligente Agenten

- **Aktionen und Verhalten**
  - Definiton „**idealer rationaler Agent**“
    - Wählt für jede Wahrnehmungsfolge immer den optimalen nächsten Schritt
    - Die Festlegung dieser Aktionen (=Agentenentwurf) ist somit eine ideale Abbildung von Wahrnehmungsfolgen auf Aktionen
    - Ein Agent kann auf eingabauten Wissen über seine Umgebung und / oder seiner Erfahrung beruhen
- **Struktur intelligenter Agenten (Agent = Architektur + Programm)**
  - Entwurf des Agentenprogramms
    - Ist eine Funktion, welche die Abbildung von Wahrnehmungsfolgen auf Aktionen implementiert
    - Die Recheneinheit, auf welcher das Programm läuft, wird Architektur genannt
      - Stellt über Sensoren Informationen zur Verfügung
      - Überträgt gewählte Aktionen auf die Effektoren
  - **Agentenprogramm**
    - Nachteile von Lookup-Programmen
      - Extrem lange Tabellen
      - Erstellen der Tabelle sehr Zeitaufwendig
      - Keinerlei Autonomie (Erfahrungsabhängigkeit) und somit unflexibel bei Umweltänderung
      - Selbst wenn er Lernfähig wäre, müsste er stetig Einträge für seine Tabelle lernen
    - **Einfache Reaktive Agent**
      - Wenn das Auto vor uns bremst, dann leite Bremsvorgang ein
      - Einfache reaktive Agenten werden auch als Reflexive Agenten bezeichnet
    - **Agenten, die die Welt beobachten** (sind auch Reflexive Agenten)
      - Beobachten Umwelt
      - Regelmäßige Aktualisierung des internen Zustandes
      - Benötigen Informationen, wie sich Umwelt unabhängig vom Agenten verändert und entwickelt
    - **Zielbasierte Agenten**
      - Um im Falle unterschiedlich möglicher Folgezustände die richtige zu wählen, benötigt der zielbasierte Agent Informationen über Ziele
      - Der Agent vergleicht diese Information mit dem möglichem Ergebnis und wählt so die optimale Aktion aus
      - Dies ist meist nicht mit einem Schritt sinnvoll möglich, sondern nur mit mehreren
        - Ermittlung solcher Atkionsfolgen über Suchen und Planen
    - **Nutzenbasierte Agenten**
      - Der Nutzen ist eine Abbildung der Menge der Zustände in die reellen Zahlen
      - Die zugeordnete Zahl beschreibt, wie günstig der Zustand für den Agenten ist
      - Um ein Ziel zu Erreichen kann der Nutzen auf zwei Arten verwendet werden
        - Wenn es Ziele gibt, die einen Konflikt bilden, lässt sich mit Nutzen Trade-off bestimmen
        - Wenn es mehrere erstrebenswerte Ziele gibt, lässt sich mit Hilfe des Nutzens die Wahrscheinlichkeit des Erreichens eines Ziels gegen seine Wichtigkeit abwiegen
  - **Umgebung**
    - **Eigenschaften von Umgebungen**
      - Zugänglich – Unzugänglich
      - Deterministisch – indeterministisch
      - Episodisch – nicht episodisch
      - Statisch – dynamisch
      - Diskret – kontinuierlich
    - **Umgebungsprogramme**
      - Stellen eine Simulationsumgebung dar, mit der Agentenprogramme getestet werden können
      - Der Simulator nimmt einen oder mehrere Agenten als Eingabe und stellt den Agenten die richtigen Wahrnehmungen zur Verfügung
      - Der Simulator nimmt die Aktionen des Agenten entgegen und Aktualisiert gegebenenfalls die Umgebung
      - Eine Umgebung ist bestimmt durch ihren Anfangszustand und ihre Update-Funktion

Kapitel 3 – Problemformulierung und Problemtypen

- **Problemlösende Agenten**
  - Zielformulierung, ausgehend von der aktuellen Situation, ist der erste Schritt beim Problemlösen
  - Ein Ziel wird als eine Menge von Weltzuständen betrachtet, in denen das Ziel erfüllt ist
  - Aktionen werden als Übergänge zwischen Weltzuständen betrachtet
  - Bei der Problemformulierung wird entschieden, welche Aktionen und welche Zustände betrachtet werden sollen. Sie folgt nach der Zielformulierung.
  - Hat ein Agent mehrere mögliche Aktionen mit unbekannten Ausgang, kann er eine Entscheidung herbeiführen, indem er verschiedene mögliche Aktionsfolgen, die zu dem gewünschten Zustand führen, untersucht und die Optimale heraussucht
    - Dieser Vorgang heißt Suche
    - Problembeschreibung als Eingabe und Aktionsfolge als Ausgabe
    - Die Lösung kann dann in der Ausführungsphase ausgeführt werden

- **Problemformulierung**
  - Um ein Problem formulieren zu können, müssen Zustände bekannt und Aktionen definiert werden
  - Problemtypen
    - **Ein-Zustands-Probem**
      - Der Agent **weiß**, in welchem Zustand er sich befindet (Umgebung zugänglich) und was jede Aktion bewirkt
      - So kann er ausgehend von seinem Zustand, für eine Aktionsfolge vorausberechnen, in welchem Zustand er sich nach der Ausführung der Aktionen befinden wird
    - **Mehr-Zustands-Probem**
      - Der Agent **weißnicht**, in welchem Zustand er sich befindet (Umgebung unzugänglich), aber er weiß was jede Aktion bewirkt.
      - Dann kann er trotzdem das Erreichen eines Zielzustandes vorausberechnen
      - Dazu muss er aber über eine Menge von Zuständen **schlussfolgern** und nicht nur über einen.
    - **Kontingenz-Problem**
      - Der Agent kann nicht im Voraus eine bestimmt Aktionsfolge berechnen, die zu einem Zielzustand führt.
      - Er benötigt während der Ausführung von Aktionen Sensorinformationen, um zu entscheiden, welche der Aktionen zum Ziel führt
      - Statt einer Folge von Aktionen braucht er also einen Baum, in dem jeder Zweig von der Wurzel zu einem Blatt eine Aktionsfolge darstellt
      - Jede Aktionsfolge behandelt ein kontingentes Ereignis
    - **Explorations-Problem**
      - Der Agent kennt die Auswirkungen seiner Aktionen nicht
      - Deshalb muss er experimentieren, um herauszufinden, was die Aktionen bewirken und welche Art von Zuständen es gibt
      - Dies kann er nur in der realen Welt tun
      - Dabei kann er aber lernen, d.h. eine interne Repräsentation der Welt aufbauen und zur Entscheidungsfindung nutzen
  - **Wohldefinierte Probleme und Lösungen**
    - Zur Definition eines Ein-Zustand-Problems sind folgende Informationen notwendig:
      - Ein Anfangszustand
      - Eine Menge möglicher Aktionen
      - Ein Zielprädikat
      - Eine Pfadkostenfunktion
    - Der Anfangszustand ist der Zustand, von dem der Agent weiß, dass er sich in Ihm befindet.
    - Eine Aktion wird als Operator oder Nachfolgefunktion bezeichnet, in Verbindung mit dem Zustand, zu dem sie von einem Gegebenen aus führt.
    - Der Anfangszustand und die Menge der Aktionen definieren den **Zustandsraum**
    - Er besteht aus einer Menge aller Zustände, die vom Anfangszustand aus durch bestimmt Aktionsfolgen, erreichbar sind.
    - Ein **Pfad** im Zustandsraum ist einfach nur eine Aktionsfolge, die von einem Zustand in einen Anderen führt.
    - Das **Zielprädikat** kann der Agent auf einen Zustand anwenden, um zu Prüfen, ob e dieser ein Zielzustand ist.
    - Die Pfadkostenfunktion ordnet jedem Pfad im Zustandsraum einen Kostenwert zu.
      - Dieser ist die Summe der Kosten der einzelnen Aktionen entlang eines Pfades.

- Der Datentyp Problem ist also definiert durch Anfangszustand, Operatoren, Zielprädikat und Pfadkostenfunktion
- Instanzen dieses Datentyps bilden die Eingaben für Suchalgorithmen.
- Die Ausgabe einer Suche ist dann eine Lösung, d.h. ein Pfad von einem Ausgangszustand in einen Zustand, welcher das Zielprädikat erfüllt

- Beim Mehrzustandsproblem werden Zustände durch Zustandsmengen und Aktionen durch Operatormengen ersetzt, welche die Folgezustandsmenge spezifizieren
  - Ein Operator wird auf eine Zustandsmenge angewandt, indem er auf jeden einzelnen Zustand angewendet wird und die Ergebnisse vereinigt werden
  - Anstelle des Ergebnisraumes erhält man somit einen **Ergebnismengenraum**

- Performance einer Problemlösung wird durch folgende Größen gemessen
  - Wird eine Lösung gefunden?
  - Wie hoch sind die Pfadkosten?
  - Wie hoch sind die Suchkosten?

  - Die **Gesamtkosten** sind die Summe aus Pfad- und Suchkosten.

- **Beispielprobleme**

  - **8-Puzzle**

    - Zustände
      - Beschreibung der Platzierung aller acht Plättchen sowie der Leerstelle
    - Operatoren
      - Bewegung der Leerstelle nach links, rechts, oben oder unten
    - Zielprädikat
      - Zustand mit gewünschter Platzierung der Plättchen
    - Pfadkosten
      - Jeder Schritt hat Kosten von 1
      - Die Kosten eines Pfades sind somit gleich der Pfadlänge.

  - **8-Damen-Problem**

Kapitel 4 – Problemlösen durch Suchen

- **Suchbäume und allgemeiner Suchalgorithmus**

  - Ein Knoten v eines Suchbaums ist eine Datenstruktur mit fünf Komponenten

    - Der Zustand des Zustandsraums, dem v entsp richt
    - Der Knoten der v erzeugt hat, also der Vater von v
    - Der zur Erzeugung von v verwendete Operator bzw. Aktion
    - Die Zahl der Knoten von der Wurzel zu v, was der Tiefe entspricht
    - Die Pfadkosten des Pfades vom Anfangszustand bis zu dem v entsprechendem Zust and

  - Zusätzlich zum Suchbaum wird eine Datenstruktur benötigt, um zwar schon erzeugte, aber noch nicht expandierte Knoten zu speichern. Diese Menge heißt **Rand** und wird als Liste implementiert und hat folgende Operationen

    - Make-List – Erzeugt eine Liste aus gegebenen Elementen
    - Emtpy – prüft ob Liste leer ist
    - Remove-Front – entfernt erste Element aus Liste
    - Listing-FN – fügt eine Menge von Elementen in die Liste ein
      - Für das Einfügen gibt es verschiedene Möglichkeiten, welche auch die verschiedenen Varianten von Suchalgorithmen darstellen

  - **Der Allgemeine Suchalgorithmus**

    - **Function** Allgemeine Suche(Problem,Listing-FN) **return** Solution or Error;
      - Nodes = Make-List (Make-Node(Initial-State[Problem]));
      - **Loop do**
        - **If** (Empty(nodes)) return Error;
        - *Node* = Remove-Front(nodes);
        - **If** (State(*node*)== Zielprädikat[problem]) return *node*;
        - Nodes=Listing-FN(nodes,Expand(*node*,Operators[Problem]))
      - **End**

- **Suchstrategien und Qualitätsmerkmale**
  - **Blinde Suchverfahren**
    - **Breitensuche**
      - Allgemeine Suche mit Listenfunktion mit Tail-Insert
      - Vollständig, denn findet immer die Lösung, falls diese existiert
      - Findet immer die Lösung mit dem kürzesten Pfad
      - Falls die Pfadkosten eine monoton wachsende Funktion der Tiefe des Suchbaums ist, dann ist BFS sogar optimal
      - Bei einem Verzweigungsfaktor von b und einer Tiefe von d liegt der Zeit- und Speicheraufwand von BFS bei $O(b^d)$
    - **Kostengesteuerte Breitensuche**
      - Es wird immer der Knoten als nächster aus dem Rand expandiert, der die geringsten Pfadkosten besitzt. Unter Bestimmten Voraussetzungen wird so eine kostengünstige Lösung gefunden.
      - Die kostengesteuerte BFS findet die kostengünstigste Lösung, wenn sich die Kosten entlang eines Pfades an keiner Stelle verringern
    - **Tiefensuche**
      - Allgemeine Suche mit Listenfunktion mit Head-Insert
      - Expandiert immer den Knoten auf der tiefstmöglichen Ebene
      - Speicherplatzaufwand ist linear b*m (m ist größte im Suchbaum vorliegende Tiefe)
      - Zeitbedarf ist $O(b^m)$, da im worst case alle Knoten expandiert werden müssen
      - Nachteil der Tiefensuche ist, dass sie in unendlichen Zweigen „hängen bleibt“
      - Deshalb ist auch nicht garantiert, dass die Tiefensuche eine optimale Lösung findet
      - Tiefensuche ist weder vollständig noch optimal
    - **Tiefenbegrenzte Tiefensuche**
      - Es wird die allgemeine Suche mit Operatoren, welche Tiefe kontrollieren implementiert
      - Zeitbedarf ist dann $O(b^l)$, wenn l die begrenzte Tiefe darstellt
    - **Suche mit iterativen Vertiefen**
      - Ist eine tiefenbegrenzte Tiefensuche, welche iterativ mehrere Tiefenschnitte l anwendet, um zu einem Ergebnis zu kommen
      - Dieses Verfahren vereinigt die Vorteile der Breitensuche mit der der Tiefensuche
      - Sie ist vollständig und optimal
      - Die Suche mit iterativen Vertiefen ist das beste blinde Suchverfahren, da es auch auf große Suchbäume mit unbekannter Tiefe anwendbar ist.
    - **Bidirektionale Suche**
      - Bei diesem Verfahren wird simultan vom Ausgangszustand vorwärts und vom Zielzustand rückwärts gesucht, bis sich die beiden Suchpfade in der Mitte treffen.
      - Mit einem Verzweigungsfaktor b und einer Lösungstiefe d ist $O(2b^{d/2})=O(b^{d/2})$
      - Folgende Voraussetzungen müssen aber gegeben sein
        - Vorgänger eines Knotens muss erzeugbar sein
        - Alle Operatoren müssen umkehrbar sein, denn dann sind die Mengen der Vorgänger mit der der Nachfolger identisch
        - Falls es mehrere Zielknoten gibt, muss diese Suche als Mehrzustandsproblem durchführbar sein. Dies ist möglich, wenn die Zielzustände explizit gegeben sind. Sind nur Zielprädikate gegeben, so wird die Definition von Zuständen schwierig.
        - Es muss schnell prüfbar sein, ob ein Knoten schon in einem anderen Teil der Suche vorkommt
        - Es muss entscheidbar sein, welche Art der Suche in beiden Richtungen ablaufen soll.
  - **Heuristische Suchfunktionen**
    - Heuristische Suchalgorithmen versuchen den Suchpfad durch „Schätzen“ zu optimieren und somit in der Laufzeit effizienter zu arbeiten.
    - Es wird nun eine Evaluierungsfunktion benötigt, welche die richtige Reihenfolge für die Abarbeitung bestimmt.
    - Die Knoten werden so geordnet, dass der mit der besten Bewertung als Erster expandiert wird.
    - **Greedy Suche**
      - Der Knoten, welcher dem Zielzustandsknoten als am „Naheliegendsten“ eingeschätzt wird, wird als Erster expandiert.
      - Eine Funktion, welche solche Kosten schätzt, nennt man Heuristikfunktion
      - Eine heuristische Funktion h(n) stellt also die Kosten des billigsten Pfades von einem Knoten n zum Zielzustand dar.

- Jede beliebige Funktion kann als Heuristik verwendet werden. Es muss nur gelten h(n) = 0, falls n ein Zielknoten ist.
- Die Greedy-Suche arbeitet nach dem Tiefensuchmuster und ist somit auch nicht optimal und nicht vollständig.
- Der Worst-Case Zeitbedarf ist $O(b^m)$, mit m als maximale Tiefe des Suchraumes.

▪ **A*-Suche**

- Durch Addition der heuristischen Funktion h und der Pfadkostenfunktion g erhält man die Funktion f(n) = g(n) + h(n).
- Dabei sind g(n) alle bisherigen Kosten und h(n) die aktuellen Kosten.
- Die A*-Suche arbeitet im Grunde wie Greedy. Es wird nur g(n) hinzuaddiert.
- Hat die Funktion h die Eigenschaft, dass die Kosten eines Pfades bis zu einem Zielknoten nicht überschätzt werden, dann ist es eine **zulässige Heuristik**.
- Die Best-First-Suche unter Verwendung von f als Evaluierungsfunktion mit zulässigem h heißt A*-Suche.
- Eine heuristische Funktion, deren Werte entlang des Pfades niemals kleiner werden, nennt man **monoton**.
- Ist eine heuristische Funktion nicht monoton, dann kann sie durch die Pfadmaximierungsgleichung (Maximum bilden) im eine Monotone umgeformt werden.
- Die A*-Suche ist optimal und vollständig, aber im Worst-Case exponentiell, da sie der Breitensuche ähnelt.

o **Speicherbegrenzte Suche**

▪ **A*-Suche mit iterativen Vertiefen – IDA***

- Arbeitet wie Suche mit iterativen Vertiefen, nur dass die Suche nicht strikt tiefenorientiert ist, sondern kostenorieniert ist.
- IDA* ist optimal und vollständig und benötigt nur Speicherplatz, der proportional zum längsten Pfad, der untersucht wird, ist. Somit ist Speicherplatzaufwand linear.

▪ **Simply Memory-Bounded A* - SMA***

- Der SMA* merkt sich Knoten, welche er schon mal expandiert hat.
- Somit wird Sucheffizienz erhöht. SMA* hat folgende Eigenschaften:
  - o Benutzt allen verfügbaren Speicher
  - o Vermeidet Zustandswiederholungen, soweit Speicher es erlaubt
  - o Ist vollständig, wenn Speicher ausreicht, um kürzesten Suchpfad zu speichern
  - o Ist optimal, wenn Speicher ausreicht, um kürzesten optimalen Lösungspfad zu speichern. Andernfalls liefert er die beste Lösung, die mit dem verfügbaren Speicher möglich ist.
  - o Er ist optimal effizient, wenn der Speicher für den gesamten Suchbaum ausreicht.
- Soll ein Knoten expandiert werden und es ist kein Speicher mehr frei, so wird ein Knoten aus der Liste entfernt. Solche entfernten Knoten heißen vergessene Knoten.
- Bevorzugt werden solche Knoten entfernt, welche keine gute Lösung versprechen, d.h. hohe f-Kosten haben.
- Im Vorgängerknoten eines entfernten Teilbaums werden aber Informationen über die Qualität des besten Pfades in dem entfernten Teilbaum abgelegt, um dadurch einen solchen Teilbaum im Falle des Falles wieder rekonstruieren zu können.

Kapitel 5 – Problemlösen durch Optimieren

- Unterschied zur Suche ist, dass bei Suche der Anfangszustand gegeben ist und die Lösung gesucht wird
- Beim Optimieren ist die Lösung schon im Anfangszustand gegeben
- Zielzustände werden über Zielfunktionen bewertet
- Durch das Optimieren wird versucht einen besseren Zielzustand zu finden
- Speichern immer nur einen Zustand

- **Bergsteigen**

  - o Stures Suchen von besseren Werten
  - o Gibt es mehrere bessere Knoten, muss via Zufallsfunktion einer selektiert werden
  - o Drei schwerwiegende Probleme entstehen dabei

    ▪ **Lokales Maximum**
      - Stößt die Suche auf ein lokales Maximum, so bleibt sie dort hängen und findet ein eventuelles globales Maximum nicht

    ▪ **Plateau**
      - Einmal auf einer Ebene sitzend, kann der Algorithmus nur raten, in welche Richtung die Suche fortgesetzt werden soll

    ▪ **Grat**
      - Man benötigt Operatoren, welche verhindern, dass die Suche entlang eines Grates hin und her pendelt
  - o Um diese Problemen aus dem Weg zu gehen, kann man die Suche durch Bergsteigen mehrfach von verschiedenen zufällig gewählten Startpunkten aus durchführen und die besten Werte nehmen

- **Simuliertes Ausglühen**
    - Die Idee ist, während der Suche in zufälligen Abständen Knoten mit schlechteren Werten auszuwählen, um auf diese Weise lokalen Maxima zu entgehen.
    - Eine Variable T beschreibt im Algorithmus eine Art Temperatur (die die Wahrscheinlichkeit von Abwärtsschritten steuert)
        - T wird bei jeder Iteration inkrementiert
        - Anfangs ist T relativ hoch und es besteht somit eine hohe Wahrscheinlichkeit des Abwärtsgehens
    - Mit richtigen Parametern findet das Ausglühen also das globale Maximum
    - Ist vollständig und optimal, wenn ein entsprechend langer Zeitraum für das Ausglühen gegeben ist

- **Genetische Algorithmen**
    - Ist aus Evolutionstheorie abgeschaut
    - Man nimmt an das Evolution optimale Pfade wählt und somit optimale Wesen hervorbringt
    - Laufen auf Populationen von Genomen (Folge von Genen = Bitstrings) ab
    - Genetische Algorithmen erzeugen in einer großen Schleife immer neue Generationen von Genomen durch Anwendung folgender Operatoren
        - Selektion (Aussondern von bestimmten Genomen zum Konstanthalten der Population)
        - Kreuzung (Auswählen der fittesten Genome und Verkettung bzw. Kombination dieser)
        - Mutation (Verändert ein oder mehrere zufällig gewählte Gene eines Genoms wodurch ein neues Genom entsteht um vor allem eventuellen lokalen Maxima zu entgehen)
    - Dabei werden folgende Schritte durchgeführt
        - Definiere die Genome und eine Fitnessfunktion (Überlebensfunktion davon abhängig) und erzeuge eine initiale Population von Genomen
        - Modifiziere die aktuelle Population durch Anwendung der drei Operatoren
        - Wiederhole den letzten Schritt solange, bis sich die Fitness nicht mehr erhöht
    - Ziel des Algorithmus ist, die Fitness der Genome zu maximieren
    - Die Fitnessfunktion (kann beliebig sein) bewertet jedes neu entstandene Genom
    - Dazu muss das Genom in seinen ursprünglichen Phänotyp (Erscheinungsbild eines Genotyps) umgewandelt werden, denn auf ihm operiert die Fitnessfunktion
    - Nachteile von genetischen Algorithmen ist die schlechte Effizienz und die Notwendigkeit einer anspruchsvollen Fitnessfunktion
    - Beliebtes Anwendungsgebiet sind die neuronalen Netze, wo es vor allem auf die Fitnessfunktion ankommt

Kapitel 6 – Agenten, die logisch schlussfolgern

- Wissensbasierter Agent
    - Erste zentrale **Komponente** ist die **Wissensbasis** (repräsentiert Menge von Fakten über die Welt)
    - Verschiedene Repräsentationen werden **Sätze** genannt
    - Sätze werden in einer Wissensrepräsentationssprache formuliert
    - Zwei grundlegende Funktionen sind notwendig:
        - TELL (Hinzufügen von Informationen)
        - ASK (Abfragen von Informationen)
    - Zweite wichtige **Komponente** ist die **Inferenzmaschine**, welche versucht Wissen zu erschließen
    - Ein Wissensbasierter Agent ist den Zustandsbasierten ähnlich
        - Wegen der Wissensbasis und den Aktionen TELL und ASK kann er aber auf drei Ebenen besser beschrieben werden
            - Wissensebene oder epistemologische Ebene (das was der Agent weiß)
            - Logische Ebene (Kodierung des Wissens in Sätzen)
            - Implementationsebene (physikalische Repräsentation der Sätze auf irgendeiner gewählten Agentenarchitektur)
        - Das initiale Programm kann durch Einfügen von Sätzen aufgebaut werden (deklarativer Ansatz)
        - Hat der Agent noch einen Lernmechanismus, dann wird er **autonom**

- **Repräsentation, Schlussfolgern und Logik**
    - Ziel der Wissensrepräsentation ist es, Wissen in einer für den Rechner verständlichen Form zu kodieren
    - Eine **Wissensrepräsentationssprache** definiert sich durch:
        - Syntax, beschreibt die korrekte Abbildung von Sätzen
        - Semantik, bestimmt die Fakten der Welt, auf die sich die Sätze beziehen

- **Logische Konsequenz und Folgerung**

  - Unterscheidung von Fakten und Repräsentationen von entscheidender Bedeutung
  - Da die Welt nicht in einen Agenten gebracht werden kann, muss der Agent auf den Repräsentationen Schlussfolgern und kann nicht direkt auf den Fakten arbeiten
  - Schlussfolgern ist also ein Prozess, welcher neue Konfigurationen aus bestehenden ableitet
  - Logische Konsequenz bzw. logische Folgerung heißt, dass man nur solche Sätze erzeugen möchte, die zwingend war sind, wenn die Ausgangssätze wahr sind
  - Man sagt „aus KB folgt logisch a“ (**Knowledge Base** und gefolgerter Satz a)

  - **Inferenzprozedur**

    - Eine Inferenzprozedur die nur logische folgende Sätze erzeugt, heißt **korrekt**
    - Eine Inferenzprozedur die alle logisch folgenden Sätze ableiten kann, heißt **vollständig**

  - **Repräsentationen**

    - Programmiersprachen
      - Besonders gut geeignet, um Algorithmen und Datenstrukturen zu formulieren
      - Ungeeignet, um unpräzise formulierte Sachverhalte zu beschreiben
      - Die Unbestimmtheit einer Aussage ist schwer formulierbar
      - BSP: in (2,2) oder (2,1) ist eine Fallgrube

    - Natürliche Sprachen
      - Sind ausdrucksstark und lassen beliebige Inhalte zu
      - Wurden aber nicht für Wissensrepräsentation entworfen sondern zur menschlichen Kommunikation
      - Bedeutung einer Aussage ist hier meist nur aus dem Kontext heraus sichtbar und oft auch nicht eindeutig.
      - Natürliche Sprachen sind also extrem Kontext- und Situationsbezogen

    - Kombination beider Sprachen
      - Wissensrepräsentationssprache muss die Vorteile beider Sprachen kombinieren
      - Wissensrepräsentationssprachen stellen eine semantische Beziehung zwischen den Sätzen und den Fakten her
      - Sie sind normalerweise kompositionale Sprachen, d.h. das die Bedeutung solcher Sätze eine Funktion der Bedeutung seiner Teile darstellt.
      - **Kompositionalität**:
        - Beziehung zwischen Semantik und Syntax verschiedener aufeinander folgender Sätze

  - **Inferenz**

    - Inferenzen sind Schlussfolgerungsverfahren
    - Korrekte Inferenzen werden Deduktion genannt
    - Eine Logik heißt **gültig** oder **notwendig wahr** genau dann, wenn er unter allen möglichen Interpretationen in allen möglichen Welten war ist
      - Bsp.: „Dame X ist bedroht oder Dame X ist nicht bedroht.“
      - Ist in der Welt Damen auf Schachbrett immer wahr, egal welche Situation vorliegt
    - Ein Satz der Logik heißt **erfüllbar** genau dann, wenn es eine Interpretation in einer Welt gibt, unter der der Satz war ist. Ansonsten ist der Satz unerfüllbar.
    - Formale Inferenz mit einer korrekten Inferenzprozedur garantiert, dass nur korrekte Sätze abgeleitet werden.

  - **Logik**

    - Allgemein besteht eine Logik aus zwei Hauptbestandteilen
      - Formale Sprache zur Beschreibung eines Sachverhalts
        - Syntax
        - Semantik
      - Beweistheorie
        - Eine Menge von Regeln zum Ableiten der logischen Folgerungen

    - In der Aussagenlogik werden ganze Aussagen oder Fakten durch einzelne Symbole repräsentiert.
    - Solche Symbole können durch boolesche Operatoren verknüpft werden

    - Die **Aussagenlogik** ist kaum zur Repräsentation von Objekten geeignet und deshalb als Repräsentationssprache ungeeignet.

    - Die **Logik erster Ordnung** trägt wesentlich mehr zur Repräsentation der Welt bei

      - Sie kann Objekte und Prädikate über Objekte stellen
      - Sie erlaubt mittels Junktoren und Quantoren Sätze über alles in dieser Welt zu formulieren

- **Aussagenlogik**
  - **Syntax**
    - Symbolvorrat der Aussagenlogik:
      - Konstanten (True,False)
      - Aussagensymbole (P,Q…)
      - Junktoren (&,|,!...)
      - Klammern (,)
    - Dabei gilt
      - Jede Konstante und jedes Symbol sind Sätze
      - Ist a ein Satz, dann ist (a) auch ein Satz
      - Aus einem oder mehreren Sätzen können mit Junktoren neue gebildet werden
        - Konjunktion zweier Konjunkte
        - Disjunktion zweiter Disjunkte
        - Implikation von Prämisse zur Konklusion
        - Äquivalenz
        - Negation
      - Ein Literal ist ein atomarer oder negiert atomarer Satz
  - **Semantik**
    - Die Bedeutung zusammengesetzter Sätze lässt sich über Wahrheitstabellen ermitteln
    - Es werden Präzedenzregeln angewendet, um Klammerungen zu sparen
    - Zu beachten ist die Implikation, welche keine kausalen Dinge beschreibt
  - **Gültigkeit und Inferenz**
    - Wahrheitstabellen zum Testen auf Gültigkeit
    - Stellt ein Beweisverfahren dar, was unabhängig von der Bedeutung der Teilaussagen ist
  - **Modelle**
    - Jede Welt, in der ein Satz unter einer bestimmten Interpretation wahr ist, ist ein Modell des Satzes unter dieser Interpretation.
  - **Inferenzregeln**
    - Beweise für Gültigkeit mit Wahrheitstabellen zeigt, dass sich immer bestimmte Muster wiederholen
    - Jedes Muster beschreibt eine Inferenzklasse
    - Jedes Muster kann durch Inferenzregeln beschrieben werden
    - Eine Inferenzregel muss einmal bewiesen werden und kann dann beliebig oft angewendet werden
    - Inferenzregeln werden in der Aussagenlogik durch einen Quotienten zweier Sätze beschrieben
      - Alle konkreten Sätze die zu den durch die Inferenzregel vorgegebenen Strukturen passen, erfüllen die Inferenzregel
      - Folgende Regeln sind gebräuchlich
        - Modus ponens oder Implikationsbeseitigung
        - Und-Beseitigung
        - Und-Einführung
        - Oder-Einführung (Oder-Beseitigung nicht möglich)
        - Beseitigung der doppelten Negation
        - Unit Resolution (wenn b nicht gilt muss a gelten)
        - Resolution (Transitivität nutzen)
      - Eine Inferenzregel ist genau dann korrekt, wenn die Konklusion immer dann wahr ist, wenn alle Prämissen wahr sind
  - **Verschiedene Eigenschaften für aussagenlogische Inferenz**
    - Das Problem der Erfüllbarkeit aussagenlogischer Sätze ist NP-vollständig
    - Monotonizität der Logik heißt, dass alle Sätze aus einer Wissensbasis, auch aus jeder Obermenge der Wissensbasis folgbar sind
    - Horn-Sätze sind Sätze, auf denen Inferenzen in polynomieller Zeit anwendbar sind

Kapitel 7– Logik erster Ordnung

- Nach der Logik erster Ordnung besteht die Welt aus Objekten, d.h. Dingen, die individuelle Identitäten und Eigenschaften haben, welche die Objekte differenzieren.
- Zwischen Objekten können Relationen bestehen
- Einige der Relationen können Funktionen sein

- **Syntax und Semantik**
  - Es gibt wie in der Aussagenlogik Sätze, welche die Welt repräsentieren
  - Sätze bestehen aus Thermen und Prädikatsymbolen
  - Terme können funktionale Ausdrücke, Variablen oder Konstanten sein
  - Aus einfachen Sätzen werden durch Junktoren und Quantoren komplexe Sätze gebaut

- **Symbole**
  - **Konstantensymbole**
    - Durch Interpretation wird festgelegt, auf welches Objekt in der Welt sich ein Konstantensymbol bezieht
    - Jedes Konstantensymbol benennt genau ein Objekt
    - Nicht alle Objekte der Welt müssen aber Namen haben
    - Mehrfachbenennungen sind zugelassen
  - **Prädikatsymbole**
    - Durch Interpretation wird festgelegt, auf welche Relation in der Welt sich ein Prädikatsymbol bezieht.
    - Eine Relation ist eine Menge von Tupeln von Objekten, die die Relation erfüllen
  - **Funktionssymbole**
    - Werden durch Interpretation auf spezielle Relationen abgebildet, die in einer bestimmten Stelle (überlicherweise die Letzte) eindeutig sind
    - Eine n-stellige Funktion kann durch eine Menge von (n+1) Tupeln definiert werden
    - Die letzte Stelle eines solchen Tupels stellt den Wert der Funktion unter den ersten n Stellen dar
  - **Terme**
    - Ist ein logischer Ausdruck der sich auf ein Objekt bezieht
    - Somit sind auch Konstanten- und Variablensymbole Terme
    - Außer Konstantensymbolen sind Ausdrücke, gebildet aus einem Funktionssymbol und einer in Klammern eingeschlossenen Folge von Termen wieder Terme.
    - Ein Term der keine Variablen enthält, heißt Grundterm
  - **Atomare Sätze**
    - Bestehen aus einem Prädikatsymbol gefolgt von einer in Klammern eingeschlossenen Liste von Termen
    - Ein atomare Satz beschreibt einen Fakt, dessen Bedeutung entweder true oder false ist.
    - Ein atomarer Satz ist wahr, wenn die Relation, auf die sich das Prädikatsymbol bezieht, zwischen den Objekten gilt, auf denen sich die Terme beziehen
  - **Zusammengesetzte Sätze**
    - Mittels Junktoren können aus Sätzen komplexere Sätze gebildet werden
    - Die Teilsätze können zusammengesetzt oder auch atomar sein
    - Die Semantik zusammengesetzter Sätze sind Wahrheitswerte und sind analog zur Aussagenlogik definiert
  - **Quantoren**
    - **Universelle Quantifizierung**
      - **Allquantor**
        - Katzen sind Säugetiere
        - = Alle Katzen sind Säugetiere
        - V x Katze(x) => Säugetier(x)
    - **Existenzielle Quantifizierung**
      - Zur Repräsentation natürlichsprachlicher Sätze, die Aussagen über einige Objekte aus einer Menge machen
      - Micky hat eine Schwester, die eine Katze ist
        - Es gibt x Schwester (x, Micky) & Katze (x)
        - Auflösung wird disjunktiv verknüpft
    - **Geschachtelte Quantoren**
      - Gleichartige Quantoren werden abkürzend zusammengefasst
      - Kommen Allquatoren und Existenzquatoren gemischt vor, dann ist Reihenfolge wichtig
  - **Quatoren und Negation**
    - Anwendung der De Morganschen Regeln
  - **Gleichheit**
    - Gleichheitszeichen kann als eine Art spezielles Prädikat verwendet werden
    - Sind t1 und t2 Terme, dann ist t1 = t2 ein Satz

- **Zwei Anwendungsbereiche für die Logik erster Ordnung**
  - **Die Verwandtschaftsdomäne**
    - In der Verwandtschaftsdomäne gibt es
      - Objekte: Personen
      - Einstellige Prädikate: Männlich, Weiblich
      - Zweistellige Prädikate: Elternteil, Geschwister, Bruder,Kind,Enkel…
      - Funktionen: Mutter,Vater
    - Sätze in einer Wissensbasis werden als Axiome betrachtet
    - Mittels Definitionen kann man aus diesen Axiomen neue Konzepte erzeugen und neue Axiome ableiten
    - Beim Aufbau einer Wissensbasis gibt es zwei Probleme
      - Reichen die Axiome und Definitionen zur Beschreibung einer Domäne aus?
      - Ist die Domände durch die Axiome und Definitionen überspezifiziert?
  - **Die Mengendomäne**
    - In der Mengendomäne gibt es
      - Konstanten: LeereMenge
      - Prädikate: Element,Teilmenge,Menge

- Funktionen: Durchschnitt, Vereinigung, Einfügen

- Es gibt acht Axiome welche die Eigenschaften einer Menge definieren

- Fragen und Antworten
  - Axiome und Fakten einer Domäne werden mittels Tell in die Knowledge Base eingefügt. (Zusicherungen)
  - Mittels Ask können dann logische Konsequenzen aus den Sätzen abgefragt werden. (Fragen oder Ziele)

- **Repräsentation von Veränderungen in der Welt**

  - Diachronische Regeln, sind Regeln, welche beschreiben, wie sich eine Welt verändert oder auch nicht verändert

  - **Der Situationskalkül**

    - Die Welt wird als eine Folge von Situationen betrachtet
    - Jede Situation stellt einen „Schnappschuss" des Zustandes der Welt zu dem Augenblick dar
    - Eine Situation wird aus einer anderen durch Aktionen erstellt
    - Situationen können benannt werden
    - Um den Wechsel einer Situation zur nächsten zu beschreiben, wird Funktion benutzt
      - Ergebnis(Aktion, Situation)
      - Ihr Wert ist eine neue Situation
      - Die Situation entsteht, wenn die Aktion auf die Situation angewendet wird
    - Aktionen, welche eine Wirkung auf die Folgesituation haben sind Wirkungsaxiome
    - Axiome, welche Sicherstellen, dass eine Wirkung über mehrere Situationen erhalten bleibt, sind Frameaxiome
    - Wirkungsaxiome und Frameaxiome zusammen, beschreiben vollständig, wie sich eine Welt aufgrund der Aktionen des Agenten entwickelt

  - **Ortsbestimmung**

    - Am Beispiel der Wumpus Welt ist die Ortsbestimmung besonders wichtig
    - Der Agent muss wissen, in welche Richtung er blickt
      - Hier einfach durch einen Winkel definiert
    - Der Agent benötigt eine einfache Landkarte, um aus einer Richtung und einer Ortsangabe eine neue Ortsangabe berechnen zu können

  - **Ableiten verborgener Eigenschaften einer Welt**

    - Synchrone Regelnsetzen verschiedene Eigenschaften desselben Weltzustandes in Relation

      - Kausale Regeln

        - Geben Richtung einer angenommenen Kausalität in einer Welt wieder
        - Systeme, die mit kausalen Regeln schlussfolgern, heißen modellbasierte Schlussfolgerungssysteme

      - Diagnostische Regeln

        - Leiten das Vorhandensein verborgener Eigenschaften direkt aus den Wahrnehmungen ab
        - Man kann aber keine so aussagekräftigen Schlüsse ziehen, wie das bei Kausalen Regeln der Fall ist

- **Zielbasierte Agenten**

  - **Präferenzen zwischen Aktionen**

    - Die Wünschbarkeit von Aktionen wird durch bestimmt Prädikate beschrieben
    - Prädikate können sein: Gut, Mittel, Riskant oder Tödlich
    - Die Auswahl einer Aktion erfolgt gemäß der Reihenfolge, welche die Prädikate auf Grund ihrer Semantik haben
    - Regeln dieser Art heißen Aktions-Wert-Regeln

  - **Bestimmen von Aktionsfolgen**

    - **Inferenz**
      - Mit geeigneten Axiomen kann man mit ASK eine Aktionsfolge aus der Wissensbasis ableiten
      - Für größere Welten wäre aber der Rechenaufwand zu groß´

    - **Suchen**
      - Mit Best-First-Suche kann ein Pfad zum Ziel bestimmt werden
      - Dazu muss wissen in eine Menge geeigneter Operatoren und Zustandsrepräsentationen umgeformt werden
    - **Planen**
      - Erfolgt mit speziellen Schlussfolgerungssystemen, die entworfen wurden, über Aktionen zu folgern

Kapitel 8– Inferenzen in der Logik erster Ordnung

- Zusätzlich zu den Inferenzregeln der Aussagenlogik gibt es in der Logik erster Ordnung noch folgende drei
  - Allquator - Beseitigung (für alle…)
  - Existenzquantor - Beseitigung (Es gibt…)
  - Existenzquantor – Einführung

- **Verallgemeinerter Modus Ponens**

  - Enthält die n+1 Prämissen und die Konklusion der Substitution
  - Sie wandelt vor Anwendung alle Sätze in die kanonische Form um
    - Sätze in kanonischer Form sind Hornsätze
    - Sind entweder atomare Sätze oder Implikationen mit einer Konjunktion atomarer Sätze auf der linken und einem einzelnen atomaren Satz auf der anderen Seite
    - Alle Sätze werden beim Einfügen in die Wissensbasis in Hornsätze umgewandelt
      - durch Anwendung der Inferenzregeln
        - Existenzquantor-Beseitigung
        - Und-Beseitigung
        - Weglassen der Allquatoren
      - Nachteil ist aber, dass nicht alle Sätze in Hornsätze überführbar sind

  - Unifikation

    - Es wird angenommen, dass es eine Prozedur UNIFY gibt, die zwei atomare Sätze p und q zu einer Substitution überführt, die die beiden Eingabesätze identisch macht
    - Konflikte durch doppelte Variablennamen werden durch Umbenennung aufgelöst
    - Falls ein Unifikator für q und p exisitert, soll UNIFY immer den allgemeinsten Unfikator liefern

- **Vorwärts- und Rückwärtsverkettung**

  - Es gibt einen Satz und man möchte diese über die Wissensbasis vergleichen

  - Vorwärtsverkettung

    - Üblicherweise durch hinzufügen eines meist atomaren Satzes zur KB
    - Als Teil der Tell-Funktion realisierbar
    - Es werden alle Implikationen betrachtet
    - Gelten auch die übrigen Teile der Antezedenzien dieser Implikation kann das Sukzedenz der Implikation zur Wissensbasis zugefügt werden

  - Rückwärtsverkettung

    - Hiermit ist es möglich, alle Antworten auf eine Frage an eine KB zu bekommen
    - Realisiert über ASK-Funktion
    - Die Rückwärtsverkettung arbeitet so
      - Prüft zunächst, ob die Antworten auf die Frage direkt von der KB geliefert werden kann
      - Dann wird nach allen Implikationen gesucht, deren Sukzedenz mit der Frage unifizierbar sind
      - Dann wird versucht, die Antezedenzien dieser Implikation wiederrum durch Rückwärtsverkettung zu beweisen (Konjunkte werden hintereinander bewiesen)
      - Im Laufe dieser Schritte werden Substitutionen aufgebaut, die ausgegeben werden

- **Resolution**: Eine vollständige Inferenzprozedur

  - Die Inferenzregel Resolution

    - Resolution-Regel in erweiterter Form
    - Es wird erlaubt, dass Disjunktionen mehrere Elemente enthalten
    - Sie wird auf Sätze der Logik erster Ordnung erweitert
      - Es muss unifiziert werden
      - Substitutionen der Ausgangssätze werden abgeleitet
    - Die Resolution Regel ist eine Erweiterung des Modus Ponens
    - Die Implikative Normalform ist eine Erweiterung der Hornform (linke Seite stimmt mit Hornform überein, aber auf rechter Seite ist Disjunktion von Atomen erlaubt)
    - Der Modus Ponens ist ein Spezialfall der Resolution-Regel in implikativer Form, welcher aber kein vollsändiges Beweissystem wie die verallgemeinerte Resolution Regel ist

  - Umformung in implikative Normalform erfolgt in acht Schritten

    - Beseitigung der Implikationen
    - Negationen nach innen ziehen
    - Umbenennung von Variablen
    - Quantoren nach außen ziehen
    - Skolemisierung (Entfernen der Existenzquantoren)
    - Distribution von Konjunktion über Disjunktion (Distributivgesetz)
    - Abflachen eingebetteter Strukturen (Assoziativgesetz)
    - Umwandlung von Disjunktionen in Implikationen

- Behandlung der Gleichheit
  - Eine Möglichkeit ist die Axiomisierung, (Eigenschaften als Äquivalenzrelation)
  - Zweite Möglichkeit ist spezielle Inferenzregel Demodulation
- Resolution-Strategien
  - Unit-Präferenz
    - Führe, wenn möglich einen Inferenzschritt mit einer Klausel, die nur aus einem Literal besteht durch
  - Set of support
    - Wähle eine Teilmenge „Set of support" der Sätze aus, und wende die Resolution Regel immer auf einen Satz an aus „Set of support" und einen anderen an und füge das Ergebnis in „Set of support" ein
  - Input Resolution und Linear Resolution
    - Wähle die Sätze zur Problembeschreibung als Eingabe und wende die Resolution Regel immer auf einen Satz aus dieser Menge und einem anderen an
    - Linear Resolution ist vollständig und ist eine Verallgemeinerung der Input Resolution
  - Subsumption
    - Entfernt alle Sätze, die von einem Satz in der Wissensbasis subsumiert werden, d.h. aus diesem Satz logisch folgen
    - Dadurch kann Suchraum so klein wie nur möglich gehalten werden

Kapitel 9 - Planen

- **Grundlegende Repräsentationen eines suchorientierten Problemlösers sind**
  - **Repräsentation von Aktionen**
    - dargestellt durch Programme
    - Aktionen erzeugen Beschreibungen der Folgezustände
  - **Repräsentation von Zuständen**
    - Gegeben ist ein vollständiger Anfangszustand
    - Da Aktionen vollständige Zustandsbeschreibungen erzeugen sind alle Zustände vollständig
    - Ein Zustand ist meistens eine einfache Datenstruktur
  - **Repräsentation von Zielen**
    - Einzige verfügbare Information über Ziel ist das Zielprädikat und heuristische Funktion
    - Die Informationen werden auf die Zustände angewandt, um deren Wünschbarkeit zu bestimmen
  - **Repräsentation von Plänen**
    - Eine Lösung ist eine Folge von Aktionen
    - Eine Suche betrachtet nur zusammenhängende ununterbrochene Folgen von Aktionen, die beim Anfangszustand beginnen

- **Drei Schlüsselideen liegen dem Planen zu Grunde**
  - **„Öffne" die Repräsentation der Zustände, Ziele und Aktionen**
    - durch Verwendung geeigneter Repräsentationsformen, wie z.B. die Logik erster Ordnung
    - Zustände und Ziele sind darin Sätze
    - Aktionen sind logische Beschreibungen von Vorbedingungen und Wirkungen
    - So werden Zustände und Aktionen miteinander verbunden
  - **Der Planer kann stets Aktionen zum Plan hinzufügen**
    - Nicht nur am Anfang, auch mitten in der Planung
    - Somit müssen Planung und Ausführung nicht streng getrennt werden
    - Durch Verschieben von nicht notwendigen Aktionen nach „hinten" wird dadurch der Verzweigungsfaktor stark reduziert
  - **Meist herrscht Unabhängigkeit von Teilmengen einer Welt**
    - Deshalb kann oft ein Ziel aus Konjunktion mehrerer unabhängiger Teilziele formuliert werden
    - Divide-and-Conquer dafür prädestiniert

- **Repräsentation in STRIPS-Sprache (Standfort Research Institute Problem Solver)**
  - **Zustände**
    - Konjunktionen funktionsfreier Grundliterale (bestehen aus Prädikaten angewandt auf Konstanten)
    - Zustandsbeschreibungen müssen nicht vollständig sein (in unzugänglicher Welt ist Zustand so gut wie nie vollständig)
  - **Ziele**
    - Konjunktionen funktionsfreier Literale (Variablen zugelassen)
  - **Aktionen**
    - **Aktionsbeschreibung** (Beschreibung dessen was Agent an Umgebung ausgibt, um zu handeln)
    - **Vorbedingung** (Konjunktion vom Atomen, die spezifiziert was vor Ausführung gelten muss)
    - **Wirkung** (Konjunktion von Literalen, die beschreibt, wie sich Situation nach Aktion verändert)

- OP(Action(GeheZu(dorthin), Precond(An(hier) and Pfad(hier, dort), Effect(An(dort) and not an (hier))
- Beim Planen sind nur Aktionen enthalten und **KEINE** Zustände

- Situationsraum und Planraum

- **Situationsraum und Planraum**
  - **Situationsraum-Planer** (betrachtet Zustände)
    - Ist ein Algorithmus, der im Raum der Situationen oder Zustände einen Pfad von einem Anfangs- zu einem Zielzustand durch Anwendung von Operationen bestimmt
    - Dabei gibt es Vorwärts- und Rückwärtsplaner
    - **Rückswärtsplaner**
      - Zustände im Allgemeinen nur partiell bestimmt
      - Information reicht aber aus, um auf Vorhängerzustände zu schließen
      - Nachteil der Rückwärtsplanung ist die Konjunktion der Literale, von denen jedes ein zu erfüllendes Teilziel darstellt
    - **Planraum** (betrachtet nicht Zustände, sondern die Aktionen)
      - Ist eine Menge partielle Pläne mit den dazugehörigen Operationen, die solche Pläne in ebensolche überführen
      - Bei Erstellung eines Plans für ein Problem beginnt man mit einem einfachen unvollständigen Plan und modifiziert diesen mit den Operatoren solange, bis er vollständig ist
      - Operatoren sind Hinzufügen eines Schrittes, das Ordnen von Schritten und das Instanziieren von Variablen
        - Verfeinerungsoperatoren
          - Beschränken die Menge der vollständigen Pläne durch Bedingungen (Constraints)
        - Alle restlichen Operatoren sind Modifikationsoperatoren
      - **Kausale Kanten**
        - Beschreiben den Zweck von Schritten in einem Plan, d.h. welche Vorbedingungen für einen Situationswechsel gelten müssen
        - Ein Planungsalgorithmus schützt kausale Kanten, dass zwischen zwei Aktionen die durch eine kausale Kante verbunden sind, keine Aktion eingeschoben werden kann, welche die Vorbedingung aufheben würde
        - Muss eine Aktion eingefügt werden, kann der Planungsalgorithmus sie entweder vor der kausalen Kante (Demotion) oder nach ih r (Promotion) einfügen

- **Planen mit partiell instanziierten Operatoren**
  - Eine Vorbedingung c stellt eine mögliche Bedrohung einer Vorbedingung c' dar, wenn es einen Unifikator gibt, so dass UNIFY(c) = ! UNIFY(c') ist.
  - Es gibt drei Wege mögliche Bedrohungen zu behandeln
    - Auflösung sofort mit einem **Gleichheitsconstraint**
      - Alle möglichen Bedrohungen werden sofort aufgelöst
      - Dabei wird eine Variable mit einem Wert so vorbelegt, dass ein Widerspruch zu einer anderen Vorbedingung nicht entstehen kann (Gleichheitsconstraint)
    - Auflösung sofort mit einem **Ungleichheitsconstraint**
      - Wie oben, nur dass eine bestimmte Belegung von Variablen ausgeschlossen wird, aber alle anderen erlaubt werden
      - Schwer zu implementieren
    - Auflösung später
      - Alle Bedrohungen werden solange ignoriert, bis sie notwendige Bedrohungen werden, d.h. ein expliziter Widerspruch vorliegt
      - Dann wird durch **Demotion** oder Promotion aufgelöst
      - Vorteil ist, dass schwache Festlegungen getroffen werden können
      - Nachteil ist, dass es schwer zu entscheiden ist, ob ein Plan auch eine Lösung darstellt

Kapitel 10 – Planen und Handeln

- Zur Behandlung des Problems der unvollständigen und inkorrekten Information gibt es zwei Möglichkeiten
  - **Bedingtes Planen**
    - Es wird ein bedingter Plan konstruiert
    - Berücksichtigt jede mögliche Situation
    - Agent stellt im Plan durch Sensoraktionen fest, welchen Teil des Plans er verfolgen soll
    - Durch die Sensoraktionen werden die Constraints im Plan geprüft

- **Ausführungsüberwachung**
  - Agent beobachtet während Ausführung was geschieht
  - Dadurch kann er feststellen, wenn etwas anders läuft als geplant
  - Der Agent kann dann durch Nachplanen versuchen, sein Ziel trotzdem zu erreichen

- **Bedingtes Planen**
  - Das Wesen bedingter Pläne
  - Zur Formulierung bedingter Pläne benötigt man bedingte Ausdrücke in der Form „If (<Condition>,<Then>,<Else>)“
  - Agent muss während Ausführung die Bedingungen auf Wahrheit prüfen können
  - Dem Agenten werden Fakten durch Wahrnehmungen bekannt, die er sich durch Sensoraktionen besorgen muss.
  - Zusätzlich zum gewöhnlichen Planungsprozess wird das Konzept des **Kontextes** eines Schritts im Plan benötigt.
    - Ist die Menge aller Bedingungen die gelten müssen, damit der Schritt ausgeführt werden kann
    - Er beschreibt im Wesentlichen den Zweig, in dem der Schritt liegt
    - Ist ein Kontext eines Schrittest festgelegt, so erben alle Nachfolgschritte diesen.
    - Zwei Schritte die in verschiedenen Kontexten liegen können nicht beide ausgeführt werden
    - Deshalb können sich solche Schritte nicht beeinflussen
    - Kontexte wichtig, um Buch zu führen, welche Schritte die Vorbedingungen welcher Anderen verletzen oder bestätigen würden
  - Laufzeitvariable sind Variablen, welche erst während der Ausführung mit Werten belegt werden

- **Vollständige Integration von Planung und Ausführung**
  - Planung und Ausführung werden im folgenden nicht mehr als getrennte Prozesse betrachtet
  - Vielmehr werden sie zu **einem situierten Planungsagenten** zusammengefasst
  - Dieser Agent wird selbst als Teil eines großen Plans betrachtet – seines Lebensplans
  - Seine Aktivitäten sind:
    - Ausführen einiger Schritte des Plans, die zur Ausführung anstehen
    - Verfeinern des Plans um eventuelle Mängel zu eliminieren
    - Verfeinern des Plans aufgrund zusätzlich gewonnener Informationen
    - Überarbeiten des Plans, aufgrund unerwarteter Veränderungen der Umwelt
  - Der Agent kann bereits mit der Planausführung beginnen, wenn der Plan noch nicht komplett berechnet wurde
  - Insbesondere wenn der Plan unabhängige Teilziele erhält, lässt sich so viel Zeit sparen

Kapitel 11 – Unsicherheit

- **Handeln unter Unsicherheit**
  - Agenten haben in Realität fast nie Zugang zur vollständigen Wahrheit über ihre Umgebung
  - Es gibt fast immer fragen, welche kategorisch beantwortet werden können
  - Deshalb muß der Agent in der Lage sein, unter Unsicherheit zu handeln

  - Unsicherheit kann auch dadurch entstehen, wenn der Agent seine Umwelt falsch interpretiert oder nur unvollständig wahrnimmt

  - **Qualifikationsproblem**

    - Viele Regeln über den Anwendungsbereich können auf Grund der zu großen Komplexität, oder einfach weil einige unbekannt sind ,unvollständig sein .

  - Die Wahl der richtigen Aktion, d.h. die rationale Entscheidung ist abhängig von der
    - Relativen Wichtigkeit verschiedener Ziele
    - Wahrscheinlichkeit, daß sie und bis zu welchem Grad sie erreicht werden

- **Behandlung von unsicheren Wissen**
  - Behandlung von unsicherem Wissen in einem Bereich, welcher einen sehr hohen Anteil an unsicheren Wissen bereitstellt, scheitert aus drei Gründen (Bsp.: Medizin)
    - Faulheit, da es viel zu Aufwendig ist, die vollständige Menge an Voraussetzungen und Konsequenzen einzugeben
    - Theoretische Unwissenheit, da das Gebiet selbst keine vollständige Theorie besitzt
    - Praktische Unwissenheit, da selbst wenn alle Regeln bekannt wären, das Risiko viel zu hoch wäre (Bsp. Patienten)

  - Die Verknüpfung von Ursachen und Diagnose ist keine logische Konsequenz
  - Deshalb können auch keine Wahr / Falsch Aussagen getroffen werden, sondern bestenfalls ein **Grad an Überzeugung**
  - Umgesetzt wird dies durch einen Wahrscheinlichkeitswert zwischen null (Wahrscheinlich falsch) und eins (wahrscheinlich wahr)
  - Sätze welche subjektiv sind oder durch Vereinbarung fest gelegt wurden, werden mit der Fuzzy-Logik behandelt

  - Die Wahrscheinlichkeit, die ein Agent einem Satz zuordnet, hängt von den Wahrnehmungen ab, die er bis zum aktuellen Zeitpunkt ermittelt hat. Dieser Sachverhalt ist die **Evidenz**.
  - Somit können sich die Wahrscheinlichkeiten einer Wissensbasis ändern, wenn mehr Evidenz gewonnen wurde.
  - Deshalb müssen alle Wahrscheinlichkeitsaussagen einen Hinweis ihre Evidenz enthalten

  - Wahrscheinlichkeiten die vor Erhalt einer Evidenz gemacht werden, heißen **a priori oder unbedingte** Wahrscheinlichkeiten, die restlichen **postpriori- oder bedingte Wahrscheinlichkeiten**

- **A Priori Wahrscheinlichkeit – unbedingte Wahrscheinlichkeit P(A)**
  - Diese Aussage kann nur getroffen werden, wenn keine andere Information in diesem Kontext vorliegt
  - Liegt eine neue Information vor, kann nur noch eine bedingte Wahrscheinlichkeit getroffen werden
- **A Postpriori Wahrscheinlichkeit – bedingte Wahrscheinlichkeit P(A | B)**
  - Sobald der Agent Evidenz über eine bisher unbekannte Aussage erhält, kann er keine unbedingte Wahrscheinlichkeit mehr zuordnen
  - Er kann nun bedingte Werte zuordnen: „die Wahrscheinlichkeit von A unter der Voraussetzung, daß B alles ist, was bekannt ist“
  - Bei der Formulierung von P(A | B) ist darauf zu achten, daß keine weiter Information C vorliegt.
  - Ist dies der Fall, so kann man nur die a postpriori Wahrscheinlichkeit P(A | B and C) angegeben werden

- **Wahrscheinlichkeitsaxiome**
  - Alle Wahrscheinlichkeiten liegen zwischen 0 und 1
  - Notwendigerweise wahre Aussagen haben Wahrscheinlichkeit 1, notwendigerweise Falsche die Wahrscheinlichkeit 0
  - Die Wahrscheinlichkeit einer Disjunktion ist definiert durch P(A oder B) = P(A) + P(B) – P(A und B)

- **Kombinierte Wahrscheinlichkeitsverteilung**
  - Ein probabilistisches Modell einer Domäne besteht aus einer Menge von Zufallsvariablen, die einzelne Werte mit bestimmten Wahrscheinlichkeiten annehmen können
  - Ein atomares Ereignis ist eine Zuordnung einzelner Werte zu allen Variablen
  - Die kombinierte Wahrscheinlichkeitsverteilung P(X1…Xn) ordnet allen möglichen atomaren Ereignissen Wahrscheinlichkeiten zu
  - Da sich atomare Ereignisse gegenseitig ausschließen, ist ihre Konjunktion zwangsweise falsch und ihre Disjunkion zwangsweise wahr.

- **Bayessche Regel**
  - Durch gleichsetzen der Produktregeln erhält man die Gleichung für die Bayessche Regel
  - Alle probabilistischen Systeme in der KI basieren auf ihr
  - Die Gleichung repräsentiert wieder eine Menge von Gleichungen, die die entsprechenden Elemente in der Ergebnistabelle zueinander in Beziehung setzen

Kapitel 13 – Lernen aus Beobachten

- **Allgemeines Modell lernender Agenten**
  - **Bestandteile**
    - **Performanzelement**
      - Wählt externe Aktionen auf Grund von Wahrnehmungen aus
      - Es steht also für das, was bisher der ganze Agent war
    - **Lernelement**
      - Ist dafür zuständig Verbesserungen vorzunehmen
      - Nimmt als Input Wissen über das Performanzelement und Rückmeldung aus der Kritik
      - Gibt Verbesserungen an das Performanzelement aus, daß sich Agent in Folgeaktionen besser verhält
    - **Kritik**
      - Beurteilt das Verhalten des Agenten und sagt dem Lernelement, wie gut sich der Agent verhält
      - Sie verwendet als Referenz einen fixen extern eingegebenen Performanzstandart, welcher nicht Teil des Agenten ist
      - Diese Referenz ist notwendig, weil die Wahrnehmungen des Agenten alleine nicht ausreichen, um Aussagen über die Qualität des Agenten treffen zu können
      - Wäre der Performanzstandard Bestandteil des Agenten, könnte er den Standard an sein Verhalten anpassen
    - **Problemgenerator**
      - Schlägt Aktionen vor, die zu neuen und informativen Erfahrungen führen
      - Diese Einheit verleit dem Agenten ein exploratives Verhalten
      - Ohne diese Einheit, würde der Agent nie etwas neues probieren, sondern immer nur Aktionen basierend auf seinem aktuellen Wissen ausführen
  - Der **Entwurf des Lernelements** orientiert sich an den folgenden vier Punkten
    - Welche Komponenten des Perfomanzelements müssen verbessert werden?
    - Welche Repräsentation wird für diese Komponenten benutzt?
    - Welche Rückmeldung ist erhältlich?
    - Welche vorab Information ist erhältlich?
  - **Komponenten des Performanzelements**
    - Direkte Abbildung von Bedingungen auf Aktionen über den aktuellen Zustand

- Hilfsmittel zur Ableitung relevanter Eigenschaften der Welt aus den Wahrnehmungsfolgen
- Informationen über Art und Weise, wie sich Welt verändert
- Informationen über Ergebnisse möglicher Aktionen des Agenten
- Nützlichkeitsinformationen, die die Wünschbarkeit von Weltzuständen anzeigen
- Action-Wert-Informationen, welche die Wünschbarkeit spezieller Aktionen in speziellen Weltzuständen anzeigen
- Ziele, die Klassen von Zuständen definieren, deren Erreichen den Nutzen des Agenten Maximieren

    - Jede dieser Komponenten kann gelernt werden, wenn der Agent das entsprechende Feedback erhält
    - Die Komponenten können unterschiedlich repräsentiert werden (logische Sätze oder probabilistische Formalismen wie Bayessche Netze)
    - Es gibt verschiedenste Lernalgorithmen, welche aber alle auf der gleichen Grundlage operieren

- **Methoden des Lernens**

    - **Überwachendes Lernen**
        - Agent kann Ergebnis seiner Aktionen beobachten oder es wird ihm mitgeteilt
        - „Lernen durch Lehrer“
    - **Reinforcement Learning**
        - Agent erhält eine Reaktion als Belohnung oder Bestrafung
        - Es wird ihm aber nicht gesagt, was die richtige Aktion ist (muß Agent selbst erforschen)
    - **Nicht überwachendes Lernen**
        - Agent erhält kein Feedback über die Wirkungen seiner Aktionen
        - Kann nur Beziehungen zwischen seinen Wahrnehmungen lernen
        - Falls er keine Nutzenfunktion hat, kann der Agent nicht lernen was er tun soll

- **Induktives Lernen**

    - Beim überwachenden Lernen bekommt der Lehrer eine Menge von Beispielen vorgelegt und soll daraus eine Funktion erlernen
    - Beispiele haben die Form (x,f(x)), mit x als Eingabe und f(x) die Ausgabe der zu lernenden Funktion
    - Eine **reine induktive Inferenz** (Induktion) besteht aus der Aufgabe:

        - Bestimme auf Basis einer Menge von Beispielen von f eine Funktion h, genannt die Hypothese, welche f approximiert
        - Im Allgemeinen können viele Hypothesen aus Beispielen generiert werden
        - Wird einer Hypothese einer anderen bevorzugt, so heißt dies ein **Bias**
        - Da es meist immer eine Großzahl konsistenter Hypothesen gibt, haben Lernalgorithmen einen **irgendwie gearteten Bias**

    - **Einfachste Form eines lernenden Agenten ist ein reflexiver lernender Agent**
        - Kann (Wahrnehmung, Aktion)-Paare erlernen
        - Grundstruktur eines solchen Agenten besteht aus
            - REFLEX-PERFORMACE-ELEMENT
            - REFLEX-LEARNUNG-ELEMENT

- **Klassifikation von Lernalgorithmen**

    - Lernalgorithmen auf Basis Logischer Sätze als Entscheidungsbäume oder Versionsraum-Ansatz
    - Lernalgorithmen auf Basis nicht linearer numerischer Funktionen als neuronale Netze
    - Lernalgorithmen auf Basis Bayesscher Netze

- **Lernen von Entscheidungsbäumen**

    - **Entscheidungsbäume als Performanzelemente**

        - Ein Entscheidungsbaum nimmt als Eingabe ein Beschreibung eines Objekt oder einer Situation
        - Ein Entscheidungsbaum gibt als Ausgabe ein Wahr oder Falsch an
        - Ein Entscheidungsbaum repräsentiert also eine boolesche Funktion
        - Repräsentation höherwertiger Funktionen aber auch möglich
        - Ein innerer Knoten entspricht einen Test auf den Wert einer der Eigenschaften
        - Ein Blattknoten entspricht einem booleschen Wert

    - **Ausdruckskraft von Entscheidungsbäumen**

        - Entscheidungsbäume repräsentieren Mengen von Implikationen
        - Können nicht beliebige Mengen von Sätzen der Logik erster Stufe repräsentieren
            - Können ja nur Aussagen über einzelne Objekte (Knoten) treffen
            - Aussagen über eine Objektmenge ist nicht möglich
        - Deshalb ist die Entscheidungsbaumsprache im wesentlichen auch aussagenlogisch
        - Die Ausdruckskraft von Entscheidungsbäumen entspricht somit genau der der Aussagenlogik
            - Jede boolesche Funktion oder aussagenlogische Satz kann als Baum dargestellt werden
            - Man braucht nur die Wahrheitstabelle zu iterieren und jede Zeile als Pfad im Baum interpretieren
        - Nachteilig ist die hohe Komplexität bei bestimmten Eingaben, welche schnell eine exponentielle Größe erreicht

- **Induktion von Entscheidungsbäumen aus Beispielen**
  - Ein Beispiel wird durch die Belegung der Attribute und des Zielprädikates spezifiziert
  - Der Wert des Zielprädikates heißt **Klassifikation** des Beispieles
  - Ist ein Wert des Beispiels wahr so ist es eine positive, sonst eine negative Klassifikation
  - Die gesamte Menge der Beispiele heißt **Trainingsmenge**
  - Bei der Erstellung eines Entscheidungsbaums soll ein Muster extrahiert werden, welches möglichst viele Fälle in knapper Form darstellen kann
    - Allgemeines Prinzip induktiven Lernens
    - Ockhams Rasiermesser
    - „Die wahrscheinlichste Hypothese ist die einfachste, die mit allen Beobachtungen konsistent ist."
  - Den kleinstmöglichen Entscheidungsbaum zu finden ist nicht lösbar
  - Aber mit DECITION-TREE-LEARNING ist es möglich, einen kleinen Baum zu finden
  - Dieser Algorithmus testet immer das wichtigste Attribut zuerst
    - Damit ist das Attribut gemeint, nach dem sich die Beispiele am meisten unterscheiden

- **Verwendung der Informationstherorie**
  - **Prinzip des Informationsgewinns**
    - Implementierung der CHOOSE-ATTRIBUTE Funktion grundlegend wichtig
    - Man benötigt eine Einteilung in gute und unnütze Attribute
    - Beste Werte sollten perfekten Attributen zugeschrieben werden und kleinste den wertlosen Attributen
    - **Information**
      - Ist im Sinne der Shannonschen Informationstheorie zu verstehen
      - In diesem Sinn ist eine Information eine Antwort auf eine Frage
      - Informativ ist eine Antwort dann, wenn sie neue Erkenntnisse bringt
    - Der Gehalt einer Information wird in **Bits** gemessen
    - Ein Bit genügt, um eine Ja / Nein Frage zu beantworten
    - Beim Entscheidungsbaum-Lernen wird die Wichtigkeit der einzelnen Attribute in Abhängigkeit der Anzahl enthaltener positiver und negativer Beispiele berechnet
    - Der Informationsgewinn aus dem Attributtest ist definiert durch die Differenz zwischen dem ursprünglichen Informationsbedarf und dem neuen (Gain(a))
    - Die in der CHOOSE-ATTRIBUT genutzte Heuristik ist genau die, die das Attribut mit dem größten Informationsgewinn selektiert
  - **Rauschen und Overfitting**
    - Ist ein allgemeines Problem und tritt nicht nur beim Entscheidungsbaum-Lernen auf
    - Wenn eine große Menge möglicher Hypothesen gegeben ist, besteht die Gefahr, daß man bedeutungslose Regelmäßigkeiten in den Daten entdeckt, welche die Laufzeit stark verschlechtern
    - Eine einfache Technik das **Overfitting** zu vermeiden ist das **Pruning**
      - Entdeckte Attribute, welche für Informationsgehalt irrelevant sind werden ignoriert
      - Dazu müssen aber irrelevante Attribute von relevanten unterschieden werden
        - Angenommen man teilt eine Menge von Beispielen mit einem irrelevanten Attribut auf
        - Die entstehenden Teilmengen haben dann in der Regel etwa die selbe Verteilung von positiven und negativen Beispielen wie die ursprüngliche Menge
        - Damit ist der Informationsgewinn annähernd null
      - Damit stellt sich aber die Frage, ab wann es sich lohnt ein Attribut zur Aufteilung einer Menge zu verwenden
      - Dazu werden **Signifikanztests** herangezogen
        - Er beginnt mit Annahme, es gebe kein zugrundeliegendes Muster (Nullhypothese)
        - Dann wird berechnet, wie weit die Daten vom vollständigen Fehlen eines Musters abweichen
        - Wenn der Grad der Abweichung statistisch unwahrscheinlich ist, dann besteht beträchtliche Evidenz für das Vorliegen eines signifikanten Musters in den Daten
      - Im Fall der Entscheidungsbäume ist die Nullhypothese, daß das gerade betrachtete Attribut irrelevant ist und so der Informationsgewinn für eine unendlich große Menge Beispielen null ist
      - Nun muß die Wahrscheinlichkeit dafür berechnet werden, daß unter Annahme der Nullhypothese eine Beispielmenge die beobachtete Abweichung von der beobachteten Verteilung der positiven und negativen Beispiele vergleicht
      - **Puring** liefert bei stark vertauschten Daten bessere Ergebnisse

- **Erweiterung der Anwendbarkeit von Entscheidungsbäumen**

  - **Für Erweiterung des Einsatzbereichs des Entscheidungsbaums-Lernens notwendig:**
    - Maßnahmen, um fehlende Daten zu ergänzen
    - Attribute mit besonderen Eigenschaften verwendbar machen

    - **Fehlende Daten**

      - Oft sind nicht alle Attributwerte für jedes Beispiel bekannt
      - Entweder nicht erfaßt oder erfaßbar
      - Zwei Probleme treten auf
        - Wie Konstruktion des Baumes, wenn bei einigen Beispielen die Werte fehlen?
        - Wie soll ein neues Beispiel klassifiziert werden, wenn eins der Testattribute nicht anwendbar ist?

    - **Attribute mit fehlenden Werten**

      - Ist Zahl der Werte eines Attributes sehr hoch, kann es passieren, daß der Informationsgehalt als sehr hoch eingeschätzt wird, obwohl dies nicht der Fall ist
      - Solche Attribute müssen mit Gain Ratio behandelt werden

    - **Attribute mit kontinuierlichen Werten**

      - Attribute wie Größe oder Gewicht haben kontinuierliche Wertebereiche
      - Um sie für Entscheidungsbaum-Lernen verwendbar zu machen, müssen die Wertebereiche diskretisiert werden
      - Dies wird meist manuell gemacht
      - Eine bessere Möglichkeit ist, die Attribute im Rahmen des Lernprozesses vorab zu untersuchen und eine sinnvolle Unterteilung des Wertebereiches zu finden